Juuso Pöllänen

Independent cloud-interface for M-bus water consumption meter system

Helsinki Metropolia University of Applied Sciences

Bachelor of Engineering

Automation technology

Thesis

30 March 2016

Metropolia

Author(s) Title Number of Pages Date	Juuso Pöllänen Independent cloud-interface for M-bus water consumption meter system 25 pages 30 March 2016
Degree	Bachelor of Engineering
Degree Programme	Automation technology
Specialisation option	Information Technology
Instructor(s)	Antti Koskinen, Marketing Manager Timo Tuominen, Lecturer

Traditionally water consumption has been monitored by water provider circling around sites to read physical meters and report readings forward. Now as information technology and automation is part of almost everything, also water consumption metering has experienced a revolution and more advanced remote reading methods have become common.

Commissioning company of this thesis has already a meter remote reading capability integrated to their automation-system. High price of required central unit limits the sales to larger projects where full building automation system is implemented. To fill this market gap the company gave the task to design and program a suitable low cost and easy to install stand-alone system.

The objective of this thesis is to develop a prototype of data collecting unit for M-bus-protocol based domestic water consumption meter system using programmable micro-computer platform. The collected data is sent and stored in a cloud service, where it is possible to be monitored by for example a water provider.

The product is targeted to be installed and configured by personnel with limited knowledge about automation systems and programming, such as a plumber who installs physical meter itself. Also the installation should require as little additional tools and programs as possible and no more instructions than a simple manual.

The installation process should not require complicated understanding of the programs and systems to be installed. The personnel working within the process should be able to install and configure the whole system as smoothly as possible.

Keywords	Raspberry Pi, M-Bus, Cloud computing, Water consumption

Contents

List of Abbreviations

GUI	Graphical user interface. Computer software to ease using or configuring the device and to act as media to deliver human commands to program.
CSS	Cascading stylesheet language. Web development language designed to ease creating unified style web sites.
HTML	Hypertext markup language. Common web language used creating layout of web pages
HTTP	Hypertext transfer protocol, used to view HTML document in browser
M-bus	Fieldbus-protocol used to communicate with metering devices
GPIO	General purpose input output- pins, freely programmable bare pins to interface with real world
XML	Extensible markup language, used to store data in human and machine readable format
SSH	Secure shell, protocol used to remotely connect raspberry pi command line
FTP	File transfer protocol, used to transfer files between two computers

1 Introduction

According to studies water consumption may vary between 60 to 270 liters per day between apartments and residents. Due to lack of a ways to measure individual amounts, the easiest way is to charge for a fixed amount. Previously water billing of the apartment buildings was based on the number of residents or the area of apartment. As a result of the increasing environmental awareness, the concern is directing the people and the governments to save energy. Large part of the energy consumed by residential apartments goes to water heating. There are technical and mechanical ways to save water but most significant savings are achieved by altering consumption habits. Residential water consumption measurement is efficiently directing people for saving water. According to a study conducted by Ministry of Environment billing based on residential water consumption measurement decreases consumption by 10 to 30 percent. Residential water measurement became compulsory for new houses and apartment buildings in 2011 and for older building plumping renovations in 2013. (1) (2)

Residential water consumption meters may be divided according to how readings are been collected. The least advanced practice is to read physical meters from stairwell or apartment just like previous shared meters. In practice residential consumption measurement is seen ineffective mostly because it is challenging to read individual meters (2). One solution could be remotely readable meters, today more new buildings are getting remote meters but still been read physically from buildings maintenance room. More advanced systems are equipped with cellular or internet connection to forward readings directly to maintenance company or water provider and even directly to billing software. (3)

The purpose of this study is to investigate remotely readable water consumption metering systems in witch an interface for automatic billing programs or consumption displays at apartment can be easily added.

1.1 Technology

The platform of the system was chosen to be Raspberry Pi B+ micro-computer due to its low price, massive community and common reliability. Raspberry Pi is basically low

power barebone computer with usually a Linux operating system. It also has general purpose input output-pins that are freely programmable and therefore may be used to communicate directly to level converter which converts M-bus 24-32v voltage to Raspberry Pi:s 3.3v level.

M-bus – meter bus is a fieldbus especially developed for remote reading of meters and power supply with one cable pair. Standard meters are individually addressable, data is provided in machine readable format and the bus is quite reliable and data transfer speed is suitable for even statistical analysis. (4) The level converter used in this project is also used in other products of the company and has been proven good and interface is suitable for Raspberry Pi.

The flow diagram in figure 1 demonstrates the phases of process. First data is collected and delivered to the level converter by M-Bus, converted to suitable voltage and delivered to the Raspberry Pi and finally sent to the cloud service.

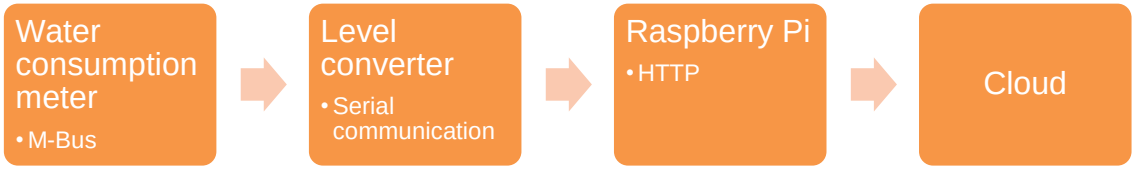

Figure 1. Physical flow of the system.

The interface was decided to be browser-based like other products of the company. Browser based user interface is easy to implement, accessible by various devices without any software installs. Therefore web-development languages are required and it is logical to implement the whole system with one. So PHP was decided to be the main programming language of this device.

2 Raspberry Pi

This chapter introduces briefly Raspberry Pi micro-computer platform and related subjects used in the project. The purpose is not to explain everything about platform

because it is not relevant to understand the development of final product. Many other micro-computers or even desktops could have been used and the project is still portable to other Linux devices with small adjustments.

2.1 History

The idea for developing a Raspberry Pi was initiated by a concern about poor programming skills of young IT professionals. At 2006 Eben Upton, Rob Mullins, Jack Lang and Alan Mycroft of University on Cambridge Computer laboratory became concerned about declining programming skills of computer science applicants. In 1990s applicants were experienced hobbyists due to less advanced computers to learn programming. At 21st century most computers have become too sophisticated and expensive to encourage programming experimentation and ICT curriculums included lessons on using productivity tools or web design rather than conventional programming. The team came up with an idea of an inexpensive computer that could boot directly to programming environment. In 2008 mobile processors had become affordable and powerful enough to provide multimedia content and gain interest from people who would not be interested in purely programming-oriented device. The project seemed realistic to carry out and a team formed Raspberry Pi Foundation to implement their plans. In 2011 the first model got into mass production and in two years it had sold over two million units. (5)

2.2 Technical specifications

The computer used in this project is Raspberry Pi B+ shown in figure 2. It is an improved version of B-model but computing power is essentially the same.

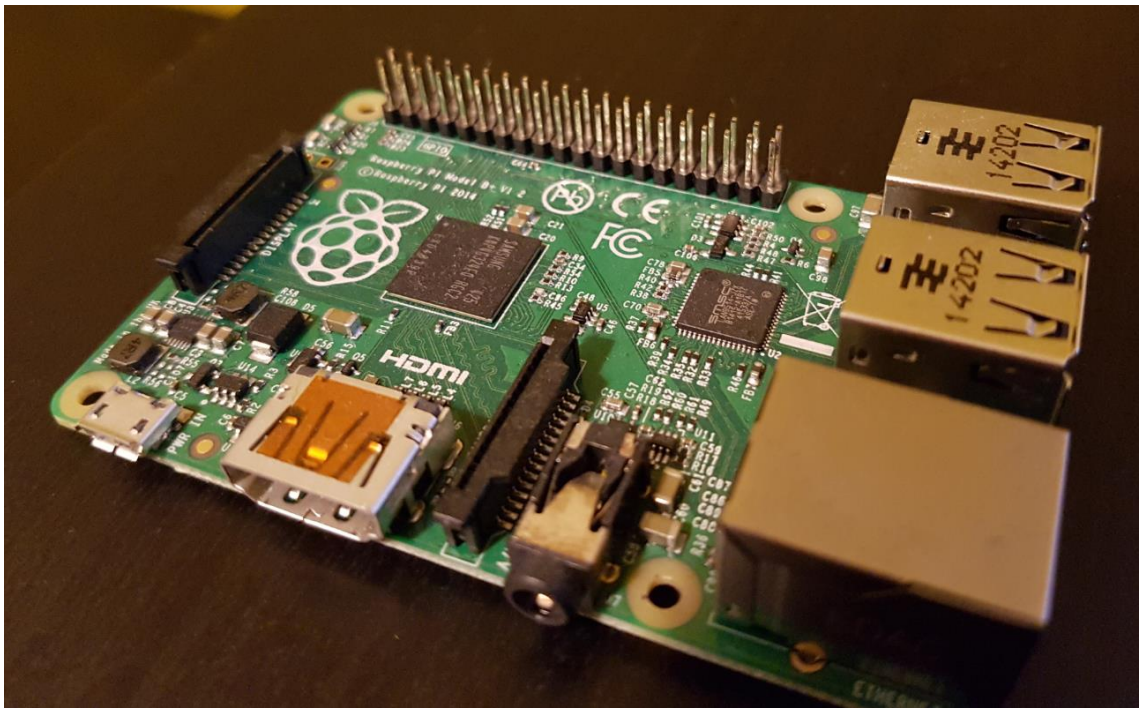

Figure 2. Raspberry Pi B+ development board

It uses Broadcom BCM2835 System-On-Chip, meaning that single chip contains most of the processor units and memory. The central processing unit is ARM11 architecture based on 700 megaherz single core unit, graphics processing unit is capable 1080p video resolution at 30 frames per second. Chip has 512 megabytes of SDRAM memory. Storage and operating system are located in a microSD- memory card. Power supply is 5 volt standard micro-USB socket and recommended current is 2 amperes. The board has Ethernet, HDMI, 3,5 mm audio jack and four USB 2.0 connectors in addition to camera and display serial interfaces and 40 pins witch 27 are General Purpose Input Output, GPIO, -pins meaning freely programmable signal transmitting or receiving ports for interfacing with real world or communicating with lower level protocols (6). (7)

2.3 Setting up

Setting up Raspberry Pi starts with operating system, there are a few different systems for different purposes, for example media center. My choice, Raspbian, is recommended by manufacturer and it is also used commonly. Desired operating system may be installed to Secure Digital memory card by downloading image-file, the exact copy of filesystem, and flashing it to memory card. The card is inserted to Raspberry and after

few configurations it is ready to use. An other option is a special software called NOOBS, abbreviation of new out of the box software. At the power up of raspberry, graphical software guides to download and install operating system via internet connection. (8 pp. 6-11)

The first configuration of plain operating system requires HDMI-display and keyboard to be connected and of course SD-card to be mounted to slot. Plugging in micro-USB plug turns Raspberry Pi on and boots operating system. Raspberry boots into command prompt where configurations like internet protocol, IP, address and other settings required to use device so called “headless”, accessing command prompt or desktop remotely from another computer with certain software without screen or keyboard. (9)

2.3.1 Raspbian

The operating system is a collection of programs and it utilities required to use hardware. Raspbian is free, Debian-linux based, operating system for Raspberry Pi hardware with still active development community provides massive amounts of pre-compiled software to increase the usability of the device. Common way to interface with the operating system is a remote terminal application as shown in the figure 3. (10)

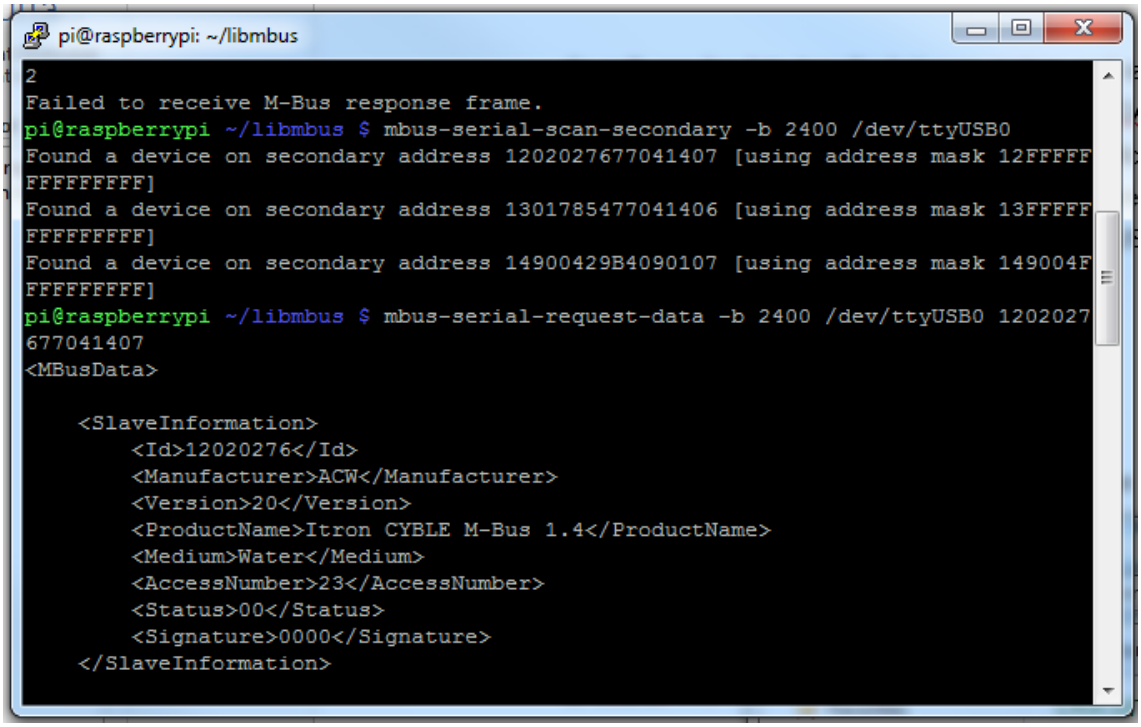

Figure 3. Raspberry Pi SSH terminal using putty

Configurations made at first boot to ease future usability start with increasing disk partition size to fill SD-card so there is more room for packages and data files. Downloadable image file is packed as small as possible to ensure compatibility with smaller cards. Raspberry can also be overclocked for better performance. The feature, called dynamic overclocking, lowers performance if a chip runs too hot to avoid damage but with sufficient cooling provides more power. Default password needs to be changed for security concerns. SSH, secure shell, is a communication protocol that allows remote access to command line interface and headless operation from another computer as demonstrated in figure 2. Enabling SSH at development phase is useful when programming and usage is made from another computer but should to be disabled when it is no more required to access user interface of operating system. Raspbian is preconfigured to request IP- address from dynamic host configuration protocol, DHCP, server if there is one in network. This means that server gives raspberry a free address from defined area when connection is established. DHCP should to be disabled and static address assigned to remotely access raspberry Pi, servers usually store certain address only for short period of time. Later IP-address is used to access created web-site. Serial connection also needs to be enabled for interfacing with physical hardware. (11) (8 pp. 14-21, 31-36)

2.3.2 Packages

Packages are Linux' way to obtain and install software used on device. Packages contain all necessary files to implement command or feature and create dependencies, pathways where to find certain feature required for operation. Packages are loaded from repository by name of program or feature by typing command *apt-get install* and name of package to command line. (12) (8 p. 72)

Apache is a common Linux web server software used to get Raspberry Pi act as web-server. Otherwise there would be no browser based interface to configure device. Apache serves hyper-text markup language, HTML, files via hyper-text transfer protocol, HTTP and is accessible from computer in same network with IP-address of Raspberry Pi. The subject is covered later, Apache simply delivers web site files stored at Raspberry to be viewed in internet-browser. Apache also requires PHP module to be able to execute

scripts that program is programmed with. Also a module for XML-processing is required. (13)

File Transfer Protocol, FTP, -server package is required to transfer files between computers when programming is performed on another computer. It is used to transfer ready web-page or script files to Raspberry. Files are transferred to dedicated folder accessible with browser by computer in same network. The development computer requires FTP-client software such as FileZilla to access Raspberry's filesystem. (14)

As software for data transfer between meters and raspberry were chosen from open source library called Libmbus version 0.8 from RaditexSCADA and raditex control AB. This library is written in C language but there is no need for modifications, supplied utility applications provide sufficient data in xml-format. The package is responsible for communication with M-bus slaves and encoding and decoding of M-bus protocol, the subject is discussed in detail later. (15)

2.4 Programming

The programming language used to program functionality was chosen to be PHP as browser based user interface requires some amount of dynamic content and there is no reason to use multiple different languages. Example of PHP programming language in the figure 4.

```
#open serials.xml and compare lists
$j=0;
$xml=simplexml_load_file("/var/www/xmldata/serials.xml") or die("Error: Cannot create object");
foreach($xml->Data->Meter as $serial) {
    $attribute = $serial->attributes();
    #echo $attribute["Id"];
    $exist = in_array($attribute["Id"], $addressesr);
    $key = array_search($attribute["Id"], $addressesr);
    $caddr[$j] = strip_tags($attribute["Id"]);
    #check if all xml serials exist in address array
    if(!$exist){
        Echo 'Serial: '.$attribute["Id"].' from apartment: '.$serial->Apartment.' appears to have communication issues.<br>';
    }
    else{
        $serial->LongId = $addresses[$key];
        #echo $addresses[$key];
        }
$j++;
}
```

Figure 4. Example of PHP code file

PHP is commonly used scripting language executed on server side, it is free to use and it works many common operating systems and servers. PHP files may contain different

web development languages like HTML, CSS, JavaScript, and of course PHP. Generally it is used to process input and output of web site to interact with user. (16)

2.5 XML

XML files plays a large part in this project as a medium to transfer and store data. XML is an abbreviation of extensible markup language. It is a markup language like HTML but intended to be used to store data in human and machine readable form instead of displaying it. XML files are manipulated with XML processor. (17)

Actually XML is not a markup language, it is a set of rules to build one. A markup language is supposed to distinguish parts of text or data and identify parts meaning and relation to each other using symbols that may be identified by reader. For example text markup contains spaces, dots, dashes and headers, XML contains tags. XML tags are certain words or symbols enclosed in chevrons like *<tag> message itself </tag>.* The symbol marks the beginning of content and a slash before a symbol marks ending. The symbols usually contains another elements. They are often quite branched and therefore it is called a tree, relations of branching tags are called child for lower level and parent for higher. The last element that has no child-elements is called leaf and usually contains data or text. Different elements that have the same parent element are called siblings. Tags may also contain additional information about tag, called attribute. Attributes are marked inside chevrons like *<tag attribute="name">.* Name "attribute" is a label of attribute and "name" is the content. Attributes are used to manipulate how certain tags behave rather than containing information itself, for example attribute label could be priority and content something prescribing like "high" or "low". (18 pp. 6, 28)

Figure 5 is example of XML-document where all basic structures are visible, root element "bookstore" contains child element "book" with three instances.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
  <book category="cooking">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  <book category="children">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="web">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```

Figure 5. An example XML-document of small bookstore database provided by w3c schools.

Books-elements have attribute category that contains information about subject of certain instance to ease browsing through larger database. Book-elements have four child elements, siblings with each other and are leaf elements that contains actual data. The title also has an attribute “lang” that contains information about language of book. The first line is a prolog containing information of document format and encoding for computer so it does not have to guess among hundreds of file formats. (18 p. 29)

3 M-Bus

M-Bus was developed to respond to the need for networking system for consumption meters at the University of Paderborn, Germany. There were lots of existing technologies for bus reading but none fulfilled requirements such as large number of connectable devices, expansion possibility, reliability, cost-effectiveness, power consumption or adequate transmission speed that metering systems set. (4) (19 p. 3)

3.1 Principles

M-bus is bus topology based on communication protocol, meaning that only one device is able to send data at the time. Bus topology offers the advantage of reliability opposing to star topology, where all devices are connected directly to central processing unit. Star topology has the advantage of communicating to all devices simultaneously or sequentially but it requires more cabling. In ring topology devices are connected to next and previous device and data is transferred from point to point circling full loop before reaching the processing unit. A disadvantage of ring topology is that one device malfunction brings the whole system down. Data is transferred at bus is in serial form, meaning that the one cable delivers bit after bit. Counterpart is a parallel transmission where bits are transferred by certain number of cables, this increases transmission speed but also cabling costs. Illustration of different topologies are shown in figure 6. (19 pp. 4-5)

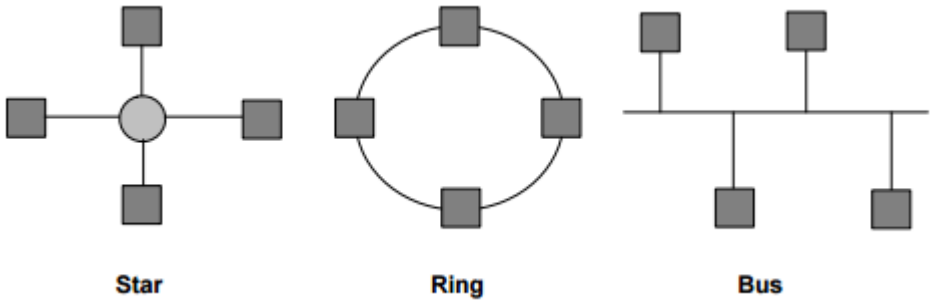

Figure 6. Different network topologies

Structure of bus topology allows transmission medium to be used by all participants. This rises a need to share lane occupation time, the method is called access technique and it is managed by an allocation logic system. Allocation logic is supposed to ensure that only one station transmits at time to avoid conflicts. Central allocation logic may share occupation time by asking sequentially for each participant's needs to transmit data, listening requests made by participants or following a predetermined timeframe. (19 pp. 5-7)

All communication has some kind of errors that need to be taken care of, simple communication protocols with small data packages utilize quite simple error recognition

methods. Usually additional parity bit is transmitted with each set of bits. A so called parity bit is set so that parity of logical ones is always odd or even, depending how it is determined in setup. Another common technique is to check character derived from message by some mathematical operation. Errors can be detected by comparing delivered character and calculated character. A faulty message is asked to be repeated to receive correct one. Faults in line or device may be detected by monitoring transmission time, if it takes too long to wait for transmission a timeout error is reported. (19 pp. 7-8)

M-bus is a hierarchical system so communication is controlled by a central allocation logic called the master. Measuring instruments, called slaves, are connected in parallel to bus-cable. To obtain large bus network with minimal costs, the power is delivered by same two-wire cable as used in communication. Data signal from master is delivered by shifting voltage levels as illustrated in figure 7. 36 v voltage represents logical “1” and 24 respectively logical “0”. Messages from slave back to master is done by alternating current consumption. Constant consumption of slaves at basic operation and presentation of logical “1” is up to 1,5 milliAmpere and 11-20 mA bump in consumption represents logical “0”. All currently used cables have some resistance and voltage delivered to slaves is less than theoretical 36 volts, therefore slaves are required to detect 12 volt drop in voltage rather than absolute 36 or 24 volts. Also the level shifter, called repeater, must adjust to actual combined current draw as each meter consumes 1,5 mA and recognize 11-20 increase in consumption as logical “0”. (19 pp. 14-15)

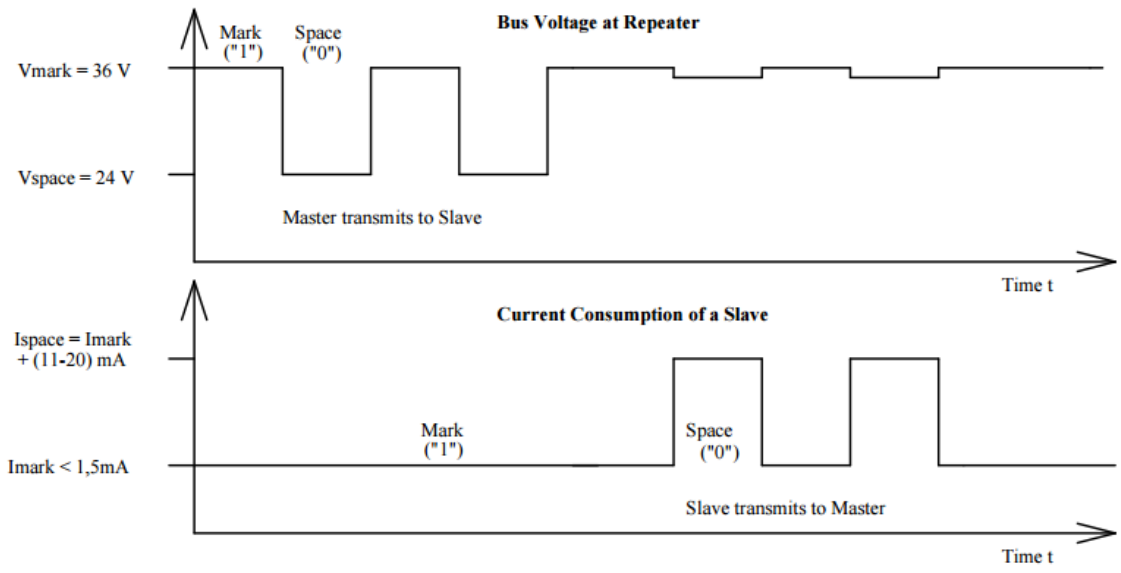

Figure 7. Illustration of voltage and current bits

The cable length of m-bus assembly is determined by cable used, with resistance of 29 ohm/km maximum cable length is 350 meters with maximum amount of 250 slaves and maximum communication speed of 9600 baud. As power supply of slaves is delivered by same cable, voltage must not drop under 12 volts in any segment or transmission of logical zero would cut operating voltage of meter to zero. The total cable length used in configuration is limited by maximum capasitance of 180 nanoFarads, in case of 180 nanoFarad per kilometer cable maximum length is one kilometer. (19 p. 16)

Normally the M-bus uses data link layer addressing, meaning that address of an individual meter is fixed and delivered via communication frame. Address consists of 8 bits and therefore, it may range from 0 to 255. Addresses 251-255 are reserved for multicast purposes, for example 254 is a test address that all meters online will answer, 255 is broadcast address that none of the meters answer. Data link layer addressing causes troubles with address allocation. Usually meters are delivered with address 0, indicating un-configured meter. All slaves require unique address from 0 to 250 defined either manually for example with DIP-switches or automatically by software. Master software send command for meters with address 0 to allocate address while connecting to the bus. Another way to address meters is extended network-layer addressing. This address consists of 8 digit identification numbers. (19 pp. 28-30, 63-65)

To streamline M-bus deployment there are slave search methods. With slaves configured with primary address it is easy to send a request to all allowed addresses with all baudrates. Secondary address slaves are not that easy, there are millions of possible identification numbers and requesting all of them would take long time and, therefore, a faster method called wildcard searching procedure was developed. Wildcard method ignores all but highest bit and sends request to all meters with the same highest bit to learn full secondary address starting from 0FFFFFFF to 9FFFFFFF. If collisions occur, it starts to change the second highest bit, for example from 50FFFFFF to 59FFFFFF and continues until no more collisions occur. By calculations made by the company Aquametro AG, illustrated in figure 8, in attempt to find 250 secondary addressed slaves, the best case scenario is to do just under 300 selections and in worst case over 7000. In average all meters are found with 1100 selections. (19 pp. 65-66, 68-69)

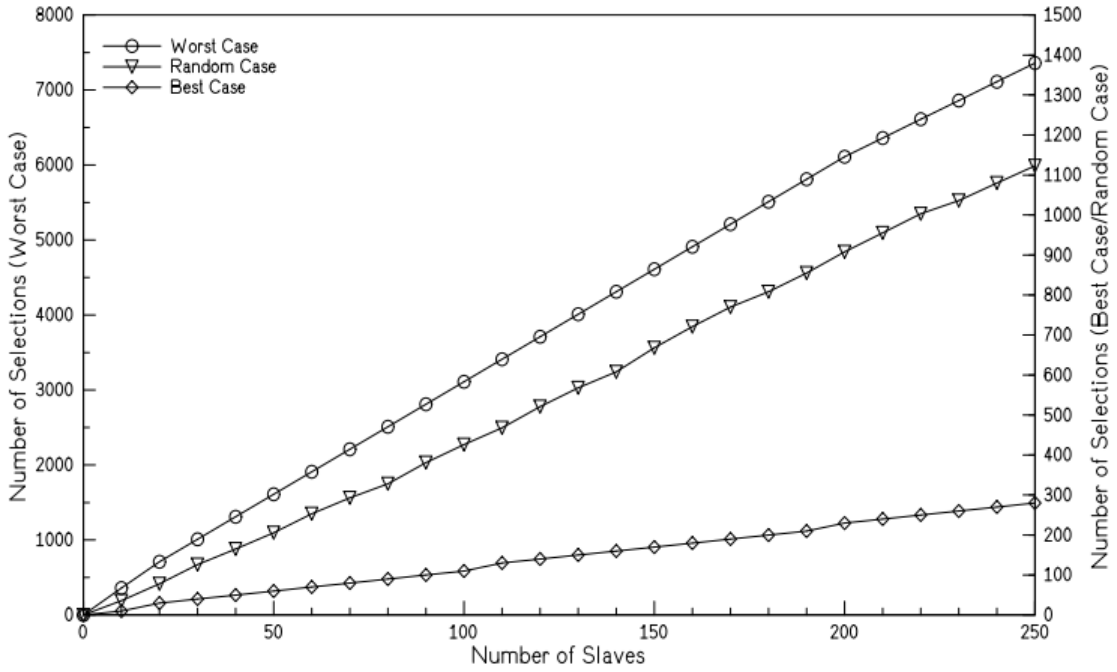

Figure 8. Wildcard search procedure selection study diagram

The identification number may be changed by master so there is a chance of two same identification numbers occurring in system. In order to enhance reliability, manufacturer specified fabrication number set during the manufacturing may be included in identification number. It is a four-byte long 8 binary coded decimal digit series running from 00000000 to 99999999 usually written to casing of meter. (19 pp. 65-66, 68-69)

3.2 M-Bus master module

As interface between meters and raspberry pi is MBUS-M13, as shown in figure 9. , level converter is manufactured by Solvimus GmbH. It is a bare circuit board with necessary components to alter voltage and detect consumption of meters and send data forward. Communication with computer unit happens through galvanically isolated TTL UART lane to protect fragile electronics from power surges caused by possible malfunctions. The converter is powered by 24VDC 0,25A power source and it is capable to support up to 80 standard meters. (20)

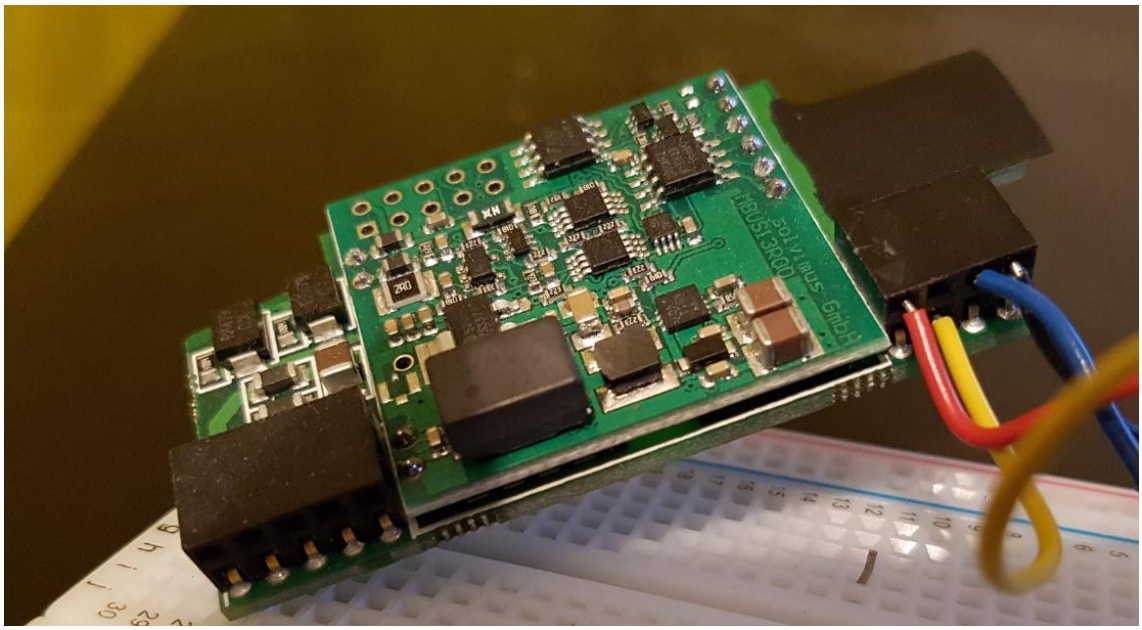

Figure 9. MBUS-M13 module

Scope of this project was not to design circuits for level converter or examine possible alternatives in greater detail. The decision to choose this particular level converter rooted from the fact that it is used in other products of the company and is proven to be reliable, not to mention there is direct hand technical assistance available. Also supply for OEM master circuits is quite limited so other options would be to design a new one or implement one from drawings found internet forums. An easy option for development purposes would have been of course to use extension module with rs-232 serial port or TCP/IP link but expenses would have been multiplied and OEM module was quite easy to connect.

4 Programming and GUI

4.1 Design

As the scope of this thesis was to provide a working prototype of water metering system master device, the main field of interest was not an actual user interface but a basic functionality and therefore not much effort was used for design. Layout of pages was made by mimicking common web-sites layout. Most web pages seem to have a navigation bar at top or side of page and I decided to place it to top due to the purpose

of the device, namely providing lists. Side navigation bar would potentially occupy too much space and render the actual table out of view in smaller screens like laptops, which are the most probable device to use in the configuration process. Figure 10 is screenshot of the user interface prototype's data editing page. Mobile devices were not considered, because again the purpose was not to design a commercial product. Future development may include support for tablets and smartphones.

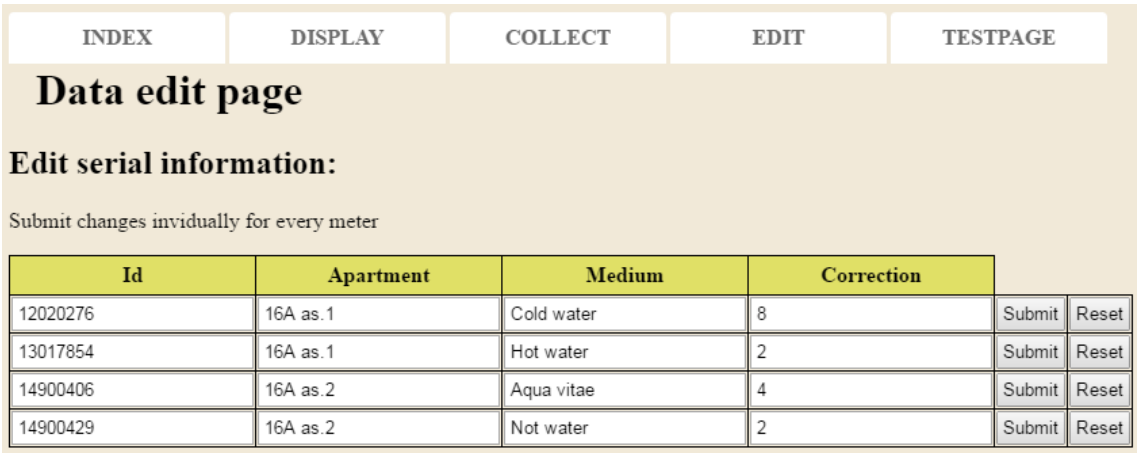

Figure 10. User interface data edit page.

However I wanted to implement a basic layout that would be easy to improve. An effective way to unify page layout was to use CSS – cascading stylesheet language. CSS is used to tell the browser how to display html documents. CSS-file containing styling instructions is linked to all HTML-documents of a web site and is read by browser during page loading. With CSS it is easy to change style of a full site instead of writing style instructions to all pages separately. As a consequence this saves lots of work when altering site style. (21)

I decided to distribute different functionalities to different pages to make the interface more clear to use and less cluttered with buttons and tables. There are three main parts in addition to log in-page: first there is an index page to provide user manual and information of device. The second page is to load configuration file, search for meters and display results. The last page is to modify a configuration file, for example input reading corrections taken from meter physical readings. Possible additional pages might be for example a maintenance page for debugging and updating device.

4.2 Features

The key feature of a device is of course the ability to read data from meters. M-bus protocol is not hard to implement and a good documentation is provided but it is still quite a task. With little scouring through the internet I found a few open-source implementations available.

The most commonly used library was Raditex Control AB:s rSCADA M-bus library implementation “Libmbus”. It is free to use as long as credit is given to the company. The library is quite versatile, it supports serial and TCP/IP-gateway communications and provides reading results in an easily processable XML-format. The software is Linux based, instructions are good and files are provided as Linux packages as well as source code. The interface is basic command line utility with necessary functions to scan and request data from primary and secondary addresses. Also community is large with multiple forum discussion about projects that utilize this library. (15)

OpenMUC jMBus is a Java based implementation of M-bus software. It is designed to support OpenMUC monitoring and logging framework and it is licenced under GPLv3. The program has capabilities to read, write, scan and listen to wired and wireless M-bus messages within terminal application. It is apparently intended to be used with Java programs and community support seems rather small. (22)

M-bus developers also have C++, C, Java and Python implementations but very old ones, the last update to site was 2005 and community support was almost non-existent (23). Other good solutions seemed to be commercial or restricted to the manufacturer’s own products.

I decided to use Libmbus library interface for data acquisition and the first phase was to get it running. Installation requires the input of a few commands and a few paths assigned in order to allow access to raspberry’s serial ports. After that commands work on command line. The problem was to make it communicate with PHP program. At first I was thinking about a sequentially executing program and reading command line output to text file, then reading text file from PHP program. However there is a function in PHP to execute command line function and read results to string or array variable directly.

The array is then processed and required data is saved to XML-file. The file contains readings from all meters and is therefore easy to process as one file.

Loads of data is not useful unless it is somehow connected to individual meters. Therefore I needed a way to connect readings to room meters. The fabrication number needs to be read from meter for identification. I decided to create a configuration file easily created with another computer for example before or during the installation of meters. As XML-file manipulation is needed in the program, it is logical to use the same functions for the configuration file. I thought that the collection of meter fabrication numbers to table with apartment numbers would be easy to do with a spreadsheet productivity tool. After some research I found out that spreadsheet programs are able to produce xml-files.

Now I had a configuration file at another computer and the challenge was to import it to raspberry. One option was to use FTP program, but it would require extra software to use the device. There are websites that offer in site option to upload files to server, so I decided to try to implement it in this one too. PHP appeared to have a function just for that and then I got an upload field to transfer the configuration file.

Manipulating two different files at same time seemed to be tricky so I decided to use the configuration file also as temporal storage for data and now there is again only one file to play with, containing all required data. Next I needed to display that data without a FTP program. PHP is able to read XML file and display data in HTML table which is an efficient way to display data in browser view.

Now I had collected and identified data which needed to be altered. Again, loading an XML file to computer is not a preferable option, so I needed a way to do it in browser. PHP and HTML forms have the capability to create a table and to input text in fields. Next, these need to be combined to get changeable table.

The most important configuration is the meter reading correction. Some meter units come with separate M-bus units like B-Meters in figure 11, used as test-device. The propeller itself may move, but the reader does not record that and has no way to know the actual reading. Therefore there has to be a correction input tool to get the same reading from the interface and a physical instrument. Another useful input is a medium,

some M-bus units know if there is cold or warm water running through but others do not. Also with some copy-paste it is easy to add possibility to correct also the room identification or serial numbers.

Figure 11. M-bus water consumption meters used to test system

After configuration, continuous operation collects data and sends it forward. Cloud service used by our company employs a HTTP POST method to receive data in XML-format which contains a few identification parameters and actual readings. PHP-language contains numerous ways to send HTTP requests. I chose to use Linux embedded cURL tool as I believe it is most reliable and the fastest way to deliver messages. Name cURL comes from "*Clients for URL's"*, it is a command line tool for file transfer using URL, *Uniform Resource Locator,* syntax, in this case XML-file containing meter readings. (24)

Figure 12 demonstrates the flow of process happening inside the Raspberry Pi. First the data is collected from the meters and delivered to the Apache in XML-format. PHP program processes the data and finally cURL transmits the data forward.

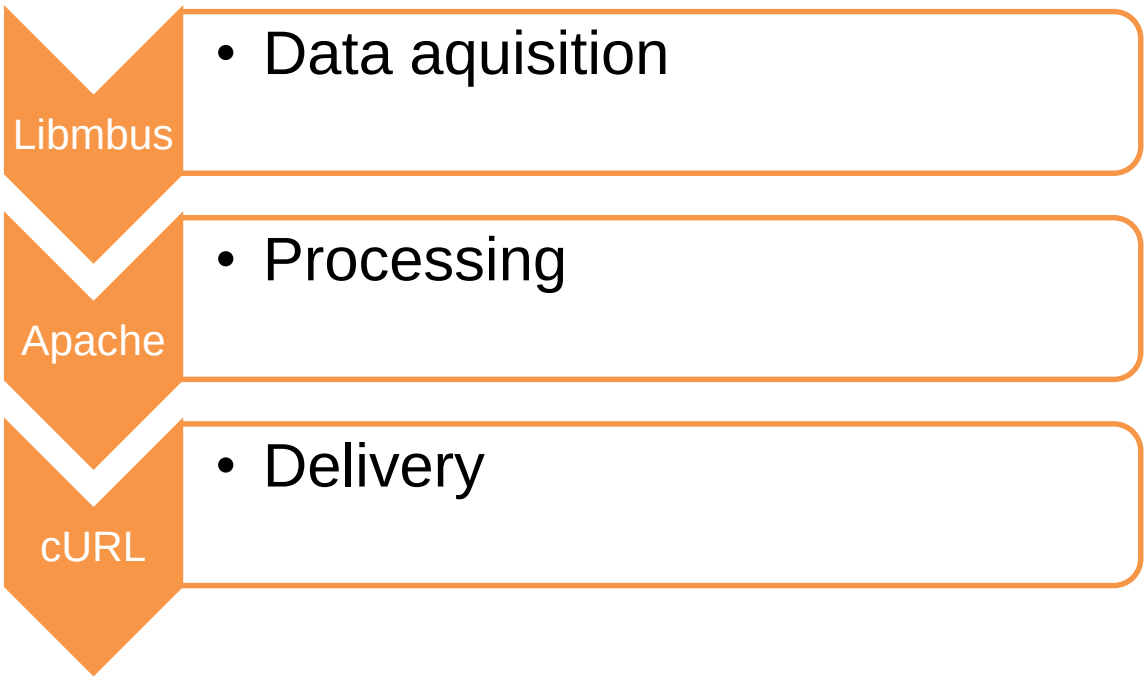

Figure 12. Process flow inside raspberry.

Further processing of data is not in scope of this thesis.

4.3 Security

As internet of things is getting more popular and the amount of connectivity rises, and therefore the importance of network security must not be ignored. Smaller and cheaper computers encourage companies to add connectivity to things where it has not been, like kettles. When rushing to be the first one to bring such devices to markets, careful engineering is often forgotten. Companies that have never worked with networking before easily ignore or do not even know about security issues. New WiFi connected water boilers with lots of wow-factor end up being a gateway to otherwise well protected systems. For example, in one occasion a certain kettle connects to any network that has the same name as one where it is supposed to be connected. Therefore a hacker may

drive past a house where one device is, steal WiFi network password, configure router to relay all traffic through malicious server and sniff rest of the passwords. (25) (26)

Whether these problems are caused by sloppy engineering or human error, it must be patched. Security updates are usually users' responsibility and no notifications are made automatically. Also specialized third party soft- and hardware is usually hard or even impossible to address by manufacturer. In Linux environment using common software, security concerns are usually found and patched quite rapidly, although the user still needs to manually check and install updates. The process could be automated for example with "*crontab*" property. Crontab is a small software where certain commands or sequences may be performed at a certain time, for example once a day. If updates to software and operating system are applied frequently the system is safe and stays that way. (26) (27 pp. 149-151)

This device contains data that may be considered personal, so security issues must not be ignored. The first line of defense is of course password protection. Apache has integrated password protection that may be used to restrict unauthorized access to browser interface or files. Different user groups may be used to restrict access of certain options, for example one allowing only configuration of networking settings and the other with full access for implementation. (28)

The next level of restriction is a device operating system, it needs to be accessed for maintenance and update purposes. Linux user interface supports multiple users and user groups with different permissions to use other files and properties. The so called "*superuser*" is the most privileged user and has a permission to access everything. The superuser should never be used in normal use to prevent accidentally altering data such as operating system files. The password should be complex to prevent malicious users obtaining more permissions than intended. Other users' permissions may be altered to allow access to necessary data only. Also software may be run by a non-privileged user. For example Apache should only require access to home directory where HTML-page files are located. In case of a successful attack, this limits damage to Apache's own home directory rather than the full operating system. (29 pp. 100-102) (27 pp. 212-216)

Raspberry Pi development is usually performed by remotely accessing command line interface with secure shell connection. Disabling this feature will improve security but

also complicate maintenance, the operating system is only accessible with monitor and keyboard. Another option is to use firewall. A property called "*IPTables*" is used to control incoming and outgoing internet traffic. IPTables contains rules what a Linux package filter does with all the packets trying to enter or leave the device. For example, if there is no requirement to access device command line remotely from anywhere else than from the network the device is connected, SSH-port may be closed from traffic from and to anywhere else, but that will not affect communication through http-port which is used to access user interface. (27 pp. 178-180)

Another useful security addition is called "*Fail2ban*". It scans traffic logs to point out suspicious addresses, for example multiple failed password attempts or exploit seeking, and add those to IPTable's restriction list. (30)

5 Conclusion

5.1 Evaluation of results

In the end the project was successful and all requirements were met. During the process there were lots of problems and obstacles to solve and the most time-consuming task was to find the best way to program a certain feature as there are loads of different ways to accomplish the same outcome. I requested for a slightly more challenging topic to learn by creating something new, instead of doing an easy task. For a next similar project or possible further development, I would use a proper integrated development environment to control files more efficiently and get rid of malfunctions due to syntax and type errors.

5.2 Project evaluation

The project itself introduced me to the world of embedded device engineering and development. As expected, it is quite a time consuming task and the phrase "90% of results are accomplished in 10% of time" was proven to be accurate. The basic platform is fast to program and the basic functionality is easy to achieve, but when a system needs

to be easy to use, reliable and still look good, tinkering and honing of details takes lots of time.

5.3 Commercialization

There is already some consumption meter systems with a cloud interface and neat graphical user interface, but usually sold as the customized product or service. Configuration requires a special tools and education to operate. Strength of this product is potential to be a commercial off the shelf product. Removal of one stage in the consumption meter installation or modernization project lowers overall costs. Affordable and easy to install remote reading system may encourage builders to implement one in a smaller projects where such has not been economic earlier, like a detached houses. Remote reading may even become de facto-standard before it is required by the regulations.

5.4 Future development

Since this study was for a bachelor's thesis, the subject had to be narrowed to include only basic functionality, in fact even the cloud service part was not supposed to be included. My personal enthusiasm would have taken it a lot further, and I wish that in future I get to continue development.

The first obvious target is to improve interface fluency. I believe the since this was first project I have ever implemented with web and server languages, the code itself is quite inefficient and crude. It needs to be cleaned and maybe some parts could be replaced with more sophisticated techniques.

Some features that could be developed include support for other types of meters, for example electricity as many of the currently popular models does have M-bus protocol implementation.

Of course circuitboard is to be put inside the casing of raspberry with an industrially manufactured connector. And some educational material or a thorough manual might be useful.

6 References

1. Talotekniikkateollisuus **ry.** teknologiateollisuus.fi. [Online] 4 2014. [Cited: 2 9, 2016.] http://talotekniikka.teknologiateollisuus.fi/sites/lvi-talotekniikka/files/file_attachments/07_VEDENMITTAUS_24032015.pdf.

2. Ympäristöministeriö**.** motiva.fi. [Online] 6 15, 2009. [Cited: 2 9, 2016.] http://www.motiva.fi/files/5725/Tyoryhmamuistio_Huoneistokohtaisten_vesimittareiden_kaytto_ja_vaikutukset_rakennusten_energiankulutukseen.pdf.

3. Kaskinen**,** Hannu**.** Taloustaito.fi. [Online] 2 12, 2014. [Cited: 2 9, 2016.] http://www.taloustaito.fi/Koti/Asuminen/Kerrostalojen-vedenkulutus-tarkkaan-syyniin/.

4. m-bus.com. [Online] [Cited: 2 9, 2016.] http://www.m-bus.com/info/mbuse.php.

5. raspberrypi.org. [Online] Raspberry Pi Foundation. [Cited: 2 10, 2016.] https://www.raspberrypi.org/about/.

6. Raspberry Pi Foundation. Raspberrypi.org - GPIO. [Online] [Cited: 2 10, 2016.] https://www.raspberrypi.org/documentation/usage/gpio/.

7. rs-components. Raspberry Pi B+ Datasheet. [Online] [Cited: 2 10, 2016.] http://docs-europe.electrocomponents.com/webdocs/1310/0900766b813109e6.pdf.

8. Monk, Simon. *Raspberry Cookbook.* s.l. : O'reilly, 2014. ISBN: 978-1-449-36522-6.

9. Raspberry Pi Foundation. Raspberrypi.org - quick start guide. [Online] [Cited: 2 10, 2016.] https://www.raspberrypi.org/help/quick-start-guide/.

10. Raspbian. raspbian.org - FAQ. [Online] [Cited: 2 10, 2016.] https://www.raspbian.org/RaspbianFAQ.

11. Raspberry Pi Foundation. Raspberrypi.org - configuration. [Online] [Cited: 2 10, 2016.] https://www.raspberrypi.org/documentation/configuration/raspi-config.md.

12. Debian.org - basics of debian package management system. [Online] [Cited: 2 10, 2016.] https://www.debian.org/doc/manuals/debian-faq/ch-pkg_basics.en.html.

13. Raspberry Pi Foundation. Raspberrypi.org - apache. [Online] [Cited: 2 10, 2016.] https://www.raspberrypi.org/documentation/remote-access/web-server/apache.md.

14. —. Raspberrypi.org - FTP. [Online] [Cited: 2 10, 2016.] https://www.raspberrypi.org/documentation/remote-access/ftp.md.

15. Raditex Control AB. rscada.se - libmbus. [Online] [Cited: 2 10, 2016.] http://www.rscada.se/libmbus/.

16. World wide web consoritum. w3schools - php. [Online] [Cited: 2 10, 2016.] http://www.w3schools.com/php/php_intro.asp.

17. w3c. XML w3c recommendation. [Online] [Cited: 3 3, 2016.] https://www.w3.org/TR/xml/.

18. Ray, Erik T. *Learning XML.* s.l. : O'Reilly, 2001. ISBN: 0-59600-046-4.

19. M.Bus Usergroup. M-Bus documentation. [Online] [Cited: 11 2, 2016.] http://www.m-bus.com/files/MBDOC48.PDF.

20. Solvimus GmbH. M-Bus master OEM module. [Online] [Cited: 2 3, 2016.] http://www.solvimus.de/en/smart-metering/m-bus-oem-master/.

21. w3c Schools. CSS Introduction. [Online] [Cited: 2 3, 2016.] http://www.w3schools.com/css/css_intro.asp.

22. Fraunhofer ISE. JMBus user guide. [Online] [Cited: 3 2, 2016.] https://www.openmuc.org/index.php?id=81.

23. Universität Bremen. Mbus software. [Online] [Cited: 3 2, 2016.] http://www.mbus.org/sw.html.

24. cUrl. cURL FAQ. [Online] [Cited: 3 24, 2016.] https://curl.haxx.se/docs/faq.html.

25. Doctorow, Cory. boingboing.net. [Online] [Cited: 3 30, 2016.] http://boingboing.net/2015/10/23/putting-your-kettle-on-the-int.html.

26. Scheier, Bruce. Wired.com. [Online] [Cited: 3 30, 2016.] http://www.wired.com/2014/01/theres-no-good-way-to-patch-the-internet-of-things-and-thats-a-huge-problem/.

27. Natarajan, Ramesh. The Geek Stuff - Linux 101 Hacks eBook. [Online] [Cited: 3 30, 2016.] http://genderandsecurity.org/sites/default/files/linux-101-hacks.pdf.

28. Apache.org. Authentication and authorization. [Online] [Cited: 3 30, 2016.] https://httpd.apache.org/docs/current/howto/auth.html.

29. Cook, Sean McManus and Mike. *Raspberry Pi for dummies.* New Jersey : John Wiley & Sons, Inc., 2013. ISBN 978-1-118-55421-0.

30. Fail2Ban.org. Fail2Ban - Wiki. [Online] [Cited: 3 30, 2016.] http://www.fail2ban.org/wiki/index.php/Main_Page.